Forecasting Student Pass Rate

UDACITY Machine Learning Engineer Nanodegree
Author: Rafal Jankowski

1. <u>**Introduction**</u>

This is classification problem since our dependent variable which we are trying to forecast is a yes/no parameter (did a student pass or not?).

2. <u>**Exploring the data**</u>

Total number of students: 395
Number of students who passed: 8215
Number of students who failed: 4030
Number of features: 30
Graduation rate of the class: 67.09%

3. <u>**Preparing the data**</u>

Results inside the file.

4. <u>**Training and evaluating models**</u>

Support Vector Machines

<u>Advantages:</u>

Effective in high dimensional spaces.

Still effective in cases where number of dimensions is greater than the number of samples.

Uses a subset of training points in the decision function, so it is also memory efficient.

Versatile – allow use of different Kernel functions.

I wanted to explore different kernel functions to see how differently they capture data complexity. Possibility of playing with different kernels and the fact that these can easily work in highly dimensional environment (given our number of features), this algorithm is a good choice for this problem. RBF kernel could be useful in capturing non-linear relationship that we see between our pass/fail data and the features.

<u>Disadvantages</u>:

Probably the biggest disadvantage of SVM is its slow speed in training and testing.

Naive Bayes

<u>Advantages:</u>
Requires a relatively small amount of data to train. It is not sensitive to irrelevant information. We have 30 features here and clearly a lot of them carry irrelevant info.

Naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering.

Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods.

Speed as well as prediction power are purposes of this exercise hence Naive Bayes should be a good competitor to something like SVM (reason number two I chose this). The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.

Disadvantages:

Naive Bayes assumes independence of features which may not always be the case.

Decision Trees

Advantages:
Decision tree classifier is the natural choice for almost any classification problem. The main reason I chose this one was that I was expecting a highly nonlinear relationships between the predictor variables.

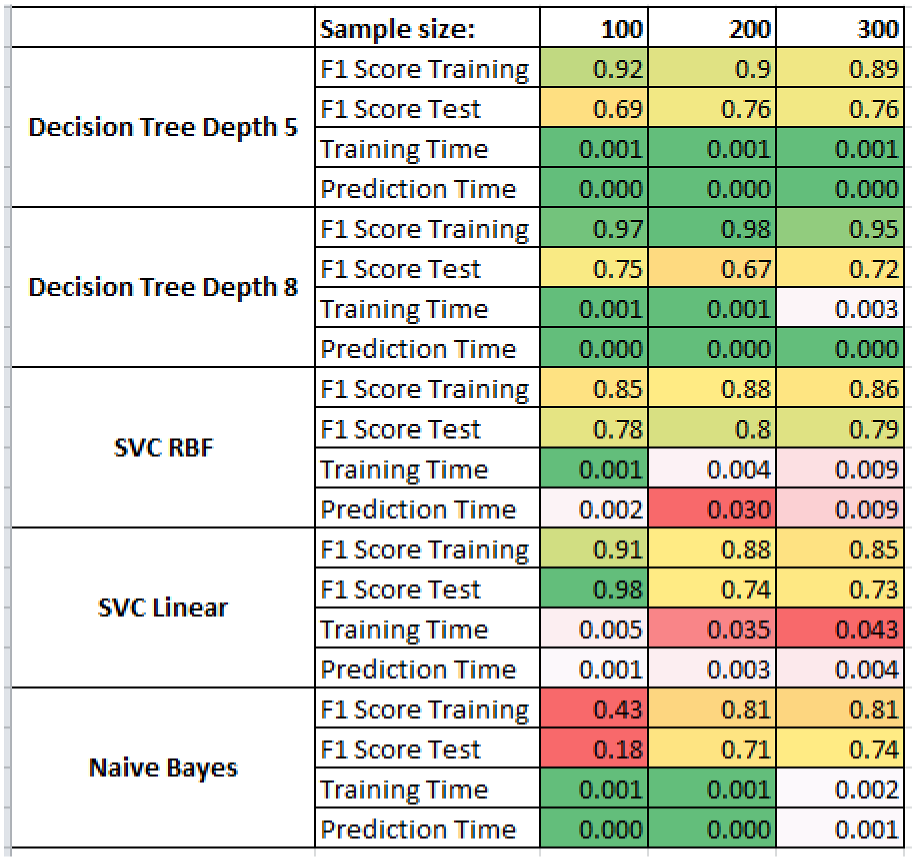
Decision tree structure however is not affected by those. I thought they could also give us quite a lot of flexibility in terms of selection depth but also are quick and easy to implement and understand.

Disadvantages:
The main problem with the decision trees is their instability.
Upon perturbing data slightly, the decision tree structure can change largely. One way around it would be to use ensemble methods in SKLEARN but were not used for this exercise.

Another potential disadvantage is that the decision boundaries tend to be parallel to the axis and clearly in real life we would expect items on the one extreme end of the boundary to behave differently to ones on the other end. Whereas in a decision tree they are all classified as identical within the branches.

5. Choosing the best model

Table below with the results of each model self-explanatory:



	Sample size:	100	200	300
Decision Tree Depth 5	F1 Score Training	0.92	0.9	0.89
	F1 Score Test	0.69	0.76	0.76
	Training Time	0.001	0.001	0.001
	Prediction Time	0.000	0.000	0.000
Decision Tree Depth 8	F1 Score Training	0.97	0.98	0.95
	F1 Score Test	0.75	0.67	0.72
	Training Time	0.001	0.001	0.003
	Prediction Time	0.000	0.000	0.000
SVC RBF	F1 Score Training	0.85	0.88	0.86
	F1 Score Test	0.78	0.8	0.79
	Training Time	0.001	0.004	0.009
	Prediction Time	0.002	0.030	0.009
SVC Linear	F1 Score Training	0.91	0.88	0.85
	F1 Score Test	0.98	0.74	0.73
	Training Time	0.005	0.035	0.043
	Prediction Time	0.001	0.003	0.004
Naive Bayes	F1 Score Training	0.43	0.81	0.81
	F1 Score Test	0.18	0.71	0.74
	Training Time	0.001	0.001	0.002
	Prediction Time	0.000	0.000	0.001

Below are % representations of the above table where the base is Decision Tree Depth 5:

	F1 score	Training Time
Decision Tree Depth 5	na	na
Decision Tree Depth 8	-5%	-2x
SVC RBF	4%	-8x
SVC Linear	-4%	-42x
Naive Bayes	-3%	-1x

SVC using RBF kernel did the best in terms of prediction power, but take considerably longer to train and apply the algorithm as do Decision Trees. In fact the F1 testing score is only 4% better than depth 5 version. Quite clearly then the winner is the Decision Tree classifier.

Next, using grid search to optimize the max depth (outside of the PY notebook file):

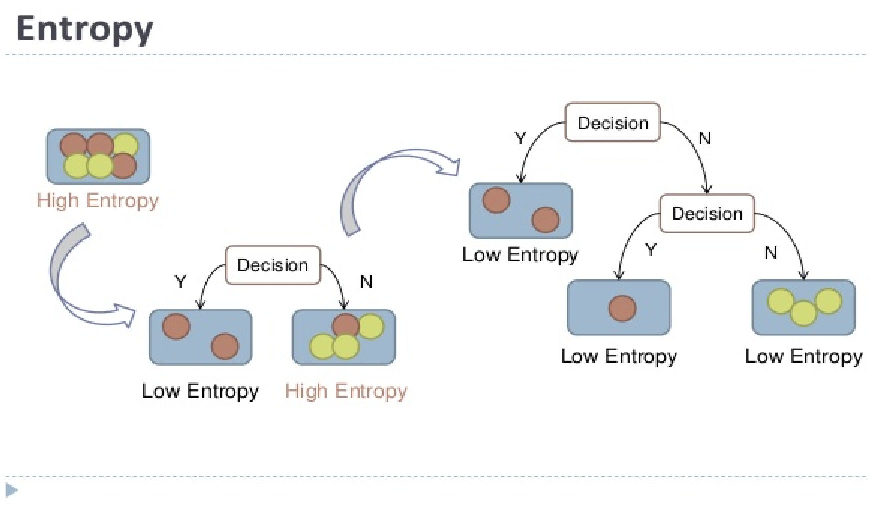
```
classifier = DecisionTreeClassifier()
parameters = {'max_depth':(1,2,3,4,5,6,7,8,9,10)}
reg = grid_search.GridSearchCV(classifier, parameters, scoring = 'f1')
reg.fit(X_train, y_train)
best_reg = reg.best_estimator_
```

Interestingly, it was at this stage where I discovered that the optimal depth for the decision tree learning in this problem was actually 1:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=1,
            max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            random_state=None, splitter='best')
```

Definition of the Decision Tree model in layman terms

A decision tree is a structure, where each internal node denotes a test on an attribute (so-called 'X-variable'), each branch represents the outcome of a test, and each leaf node holds a label which is essentially our 'Y-variable'. A test on an internal node splits further data into two sub-trees.
For example if we have student's age as one of the predictor variables, a node can split age <18 to the left and >=18 to the right. Now all of the data which is below that point will be split accordingly. The topmost node in a tree is the root node. Algorithms for constructing decision trees usually work top-down, by choosing a variable at each step that best splits the set of items. The best split is defined as one which maximizes separation of the data using term of entropy.
In other words this means that the data at each branch from the top is chosen which gives reasonably best 'split'. As an example look at these two structures:



Imagine a mixing bowl with 6 balls: 3 red and 3 yellow as per picture above. The balls have different physical properties such as temperature, surface, thickness, etc. Your task is to, in a series of decisions, be able to distinguish between objects which are red and yellow without being able to look inside the bowl prior to taking out the object.
You can ask questions such as: is the object warm or cold? And suppose that is the first decision you see on the left hand side which results in us separating 2 red balls from the pack. We would say these have low entropy, i.e. they are well separated or in our case perfectly separated and no more game is required here.
However, there is still some objects mixed up in the bowl, so we continue by asking another question. Let's say we check if the surface is smooth or rough. This time we see that the only rough ball is the red one and we have managed to separate our red balls from the yellow balls in two steps.
In reality of course a complete separation may not always be possible, so we choose structure in such a way which maximizes the split going from top to bottom.

6. <u>Supplementary: Results from the best selected model Decision Tree Classifier max depth 1:</u>

Clearly we can see that this model beats others above with testing F1 score of 84% on a 300 training size and training as well as testing times are still top of the peer group.

```
Training set size: 300
Training DecisionTreeClassifier...
Done!
Training time (secs): 0.002
Predicting labels using DecisionTreeClassifier...
Done!
Prediction time (secs): 0.000
F1 score for training set: 0.800915331808
Predicting labels using DecisionTreeClassifier...
Done!
Prediction time (secs): 0.000
F1 score for test set: 0.842857142857
```